Slip 1

Q1]

```
public class AlphabetDisplay {
    public static void main(String[] args) {
        AlphabetPrinter printer = new AlphabetPrinter();

        Thread thread = new Thread(printer);
        thread.start();
    }
}
class AlphabetPrinter implements Runnable {
    @Override
    public void run() {
        for (char ch = 'A'; ch <= 'Z'; ch++) {
            System.out.print(ch + " ");
            try {
                Thread.sleep(2000); // Sleep for 2 seconds
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}
```

Slip 2

Q2]

```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class RequestInfoServlet extends HttpServlet {
   protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
      response.setContentType("text/html");
      PrintWriter out = response.getWriter();

      String clientIPAddress = request.getRemoteAddr();
      String userAgent = request.getHeader("User-Agent");

      String serverOS = System.getProperty("os.name");
      String[] loadedServlets =
getServletContext().getServletRegistrations().keySet().toArray(new String[0]);

      out.println("<html>");
      out.println("<head><title>Request Information</title></head>");
      out.println("<body>");
      out.println("<h1>Client Information:</h1>");
      out.println("<p>IP Address: " + clientIPAddress + "</p>");
      out.println("<p>User Agent: " + userAgent + "</p>");
      out.println("<h1>Server Information:</h1>");
      out.println("<p>Operating System: " + serverOS + "</p>");
      out.println("<p>Loaded Servlets:</p>");
      out.println("<ul>");
      for (String servlet : loadedServlets) {
         out.println("<li>" + servlet + "</li>");
      }
      out.println("</ul>");
      out.println("</body>");
      out.println("</html>");
   }
}
```

Slip 3

Q2]

```
import java.util.LinkedList;
import java.util.ListIterator;

public class LinkedListOperations {
   public static void main(String[] args) {
      LinkedList<String> linkedList = new LinkedList<>();

      linkedList.add("Apple");
      linkedList.add("Banana");
      linkedList.add("Orange");
      linkedList.add("Grapes");


      System.out.println("Original LinkedList: " + linkedList);

      // ii. Delete the first element of the list
      linkedList.removeFirst();

      // Display the LinkedList after removing the first element
      System.out.println("LinkedList after removing first element: " + linkedList);

      // iii. Display the contents of the list in reverse order
      System.out.println("LinkedList in reverse order:");
      ListIterator<String> iterator = linkedList.listIterator(linkedList.size());
      while (iterator.hasPrevious()) {
         System.out.println(iterator.previous());
      }
   }
}
```

Slip 4

Q1]

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
public class BlinkText extends JFrame implements Runnable {
   private JLabel label;
   public BlinkText() {
      super("Blinking Text");
      setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      setSize(300, 100);
      setLocationRelativeTo(null);
      label = new JLabel("Blinking Text");
      label.setFont(new Font("Arial", Font.BOLD, 18));
      add(label, BorderLayout.CENTER);

      Thread thread = new Thread(this);
      thread.start();
   }
   @Override
   public void run() {
      try {
         while (true) {
            label.setVisible(!label.isVisible());
            Thread.sleep(500);
         }
      } catch (InterruptedException e) {
         e.printStackTrace();
      }
   }

   public static void main(String[] args) {
      SwingUtilities.invokeLater(new Runnable() {
         @Override
         public void run() {
            BlinkText blinkText = new BlinkText();
            blinkText.setVisible(true);
         }
      });
   }
}
```

Slip 5

Q1]

```
import java.util.Enumeration;
import java.util.Hashtable;

public class StudentHashTable {
   public static void main(String[] args) {
      Hashtable<String, String> studentDetails = new Hashtable<>();

      studentDetails.put("John", "1234567890");
      studentDetails.put("Alice", "9876543210");
      studentDetails.put("Bob", "4567890123");

      System.out.println("Student Details:");
      Enumeration<String> studentNames = studentDetails.keys();
      while (studentNames.hasMoreElements()) {
         String studentName = studentNames.nextElement();
         String mobileNumber = studentDetails.get(studentName);
         System.out.println("Name: " + studentName + ", Mobile Number: " +
mobileNumber);
      }
   }
}
```

Slip 6

Q2] import java.util.Scanner;

```
import java.util.TreeSet;

public class IntegerCollection {
   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);

      System.out.print("Enter the number of integers to input: ");
      int n = scanner.nextInt();

      TreeSet<Integer> integerSet = new TreeSet<>();

      System.out.println("Enter the integers:");
      for (int i = 0; i < n; i++) {
         int num = scanner.nextInt();
         integerSet.add(num);
      }

      System.out.println("Integers in sorted order:");
      for (int num : integerSet) {
         System.out.println(num);
      }

      System.out.print("Enter the element to search for: ");
      int searchElement = scanner.nextInt();
      if (integerSet.contains(searchElement)) {
         System.out.println("Element " + searchElement + " is present in the collection.");
      } else {
         System.out.println("Element " + searchElement + " is not present in the
collection.");
      }

      scanner.close();
   }
}
```

Slip 7

Q1]

```
import java.util.Random;
public class NumberProcessor {
   public static void main(String[] args) {
      NumberGenerator numberGenerator
= new NumberGenerator();
      SquareCalculator squareCalculator =
new
SquareCalculator(numberGenerator);
      CubeCalculator cubeCalculator = new
CubeCalculator(numberGenerator);
      Thread generatorThread = new
Thread(numberGenerator);
      Thread squareThread = new
Thread(squareCalculator);
      Thread cubeThread = new
Thread(cubeCalculator);
      generatorThread.start();
      squareThread.start();
      cubeThread.start();
   }
}
class NumberGenerator implements
Runnable {
   private Random random = new
Random();
   @Override
   public void run() {
      try {
         while (true) {
            int number =
random.nextInt(100);
            System.out.println("Generated
number: " + number);
            Thread.sleep(1000);
         }
      } catch (InterruptedException e) {
         e.printStackTrace();
      }
   }
}
class SquareCalculator implements
Runnable {
   private NumberGenerator
numberGenerator;

   public
SquareCalculator(NumberGenerator
numberGenerator) {
      this.numberGenerator =
numberGenerator;
   }

   @Override
   public void run() {
      try {
         while (true) {
            int number =
numberGenerator.getNumber();
            if (number % 2 == 0) {
          int square = number * number;
               System.out.println("Square of
" + number + " is: " + square);
            }
            Thread.sleep(1000);
         }
      } catch (InterruptedException e) {
         e.printStackTrace();
      }
   }
}
class CubeCalculator implements
Runnable {
   private NumberGenerator
numberGenerator;

   public
CubeCalculator(NumberGenerator
numberGenerator) {
      this.numberGenerator =
numberGenerator;
   }
```

Slip 8

Q1] public class TextPrinter extends Thread {
```
    private String text;
    private int times;

    public TextPrinter(String text, int times) {
        this.text = text;
        this.times = times;
    }

    @Override
    public void run() {
        for (int i = 0; i < times; i++) {
            System.out.println(text);
        }
    }

    public static void main(String[] args) {
        Thread thread1 = new TextPrinter("COVID19", 10);
        Thread thread2 = new TextPrinter("LOCKDOWN2020", 20);
        Thread thread3 = new TextPrinter("VACCINATED2021", 30);

        thread1.start();
        thread2.start();
        thread3.start();
    }
}
```

Slip 9

Q1] import javax.swing.*;

```
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
public class BallMovement extends
JFrame {
   private JPanel panel;
   private JButton startButton;
   private BallThread ballThread;
   public BallMovement() {
      setTitle("Ball Movement");
      setSize(400, 400);
setDefaultCloseOperation(JFrame.EXIT_O
N_CLOSE);
      setLocationRelativeTo(null);
      panel = new JPanel() {
         private int yPos = 0;
         @Override
         protected void
paintComponent(Graphics g) {
            super.paintComponent(g);
            g.setColor(Color.RED);
            g.fillOval(175, yPos, 50, 50); //
Draw the ball
         }
         public void moveBall() {
            yPos += 5; // Move the ball
down by 5 pixels
            repaint(); // Repaint the panel to
show the new position of the ball
         }
      };
      startButton = new JButton("Start");
      startButton.addActionListener(new
ActionListener() {
         @Override
         public void
actionPerformed(ActionEvent e) {
            if (ballThread == null ||
!ballThread.isAlive()) {
               ballThread = new BallThread();
               ballThread.start();
            }
         }
      });
      getContentPane().setLayout(new
BorderLayout());
      getContentPane().add(panel,
BorderLayout.CENTER);
      getContentPane().add(startButton,
BorderLayout.SOUTH);
   }
  private class BallThread extends Thread
{
      @Override
      public void run() {
         while (true) {
            panel.moveBall(); // Move the
ball
            try {
               Thread.sleep(100); // Delay to
control the speed of the ball
            } catch (InterruptedException e)
{
               e.printStackTrace();
            }
         }
      }
   }
   public static void main(String[] args) {
      SwingUtilities.invokeLater(new
Runnable() {
         @Override
         public void run() {
            new
BallMovement().setVisible(true);
         }
      });
   }
}
```

Slip 10

Q1]

```
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

import java.time.LocalDate;

@SpringBootApplication
public class CurrentDateApplication {

    public static void main(String[] args) {
        SpringApplication.run(CurrentDateApplication.class, args);

        // Get the current date
        LocalDate currentDate = LocalDate.now();

        // Display the current date
        System.out.println("Current Date: " + currentDate);
    }
}
```

Slip 11

Q1]

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Customer Search</title>
</head>
<body>
    <h2>Search Customer Details</h2>
    <form action="SearchServlet" method="get">
        <label for="customerNumber">Enter Customer Number:</label><br>
        <input type="text" id="customerNumber" name="customerNumber"
required><br><br>
        <input type="submit" value="Search">
    </form>
</body>
</html>
```

Slip 12

Q1]

```
import javax.swing.*;
import
javax.swing.table.DefaultTableModel;
import java.awt.*;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public class ProjectTableDisplay extends
JFrame {
   private JTable table;
   private DefaultTableModel model;
   public ProjectTableDisplay() {
      setTitle("Project Table Details");
      setSize(600, 400);
setDefaultCloseOperation(JFrame.EXIT_O
N_CLOSE);
      setLocationRelativeTo(null);
      model = new DefaultTableModel();
      table = new JTable(model);
table.setAutoResizeMode(JTable.AUTO_R
ESIZE_ALL_COLUMNS);
      JScrollPane scrollPane = new
JScrollPane(table);
      getContentPane().add(scrollPane,
BorderLayout.CENTER);
      fetchProjectDetails();
   }
   private void fetchProjectDetails() {
      try {
         Connection connection =
DriverManager.getConnection("jdbc:mys
ql://localhost:3306/your_database",
"your_username", "your_password");
         Statement statement =
connection.createStatement();
         ResultSet resultSet =
statement.executeQuery("SELECT *
FROM PROJECT");
         int columnCount =
resultSet.getMetaData().getColumnCount
();
         for (int i = 1; i <= columnCount; i++)
{
model.addColumn(resultSet.getMetaDat
a().getColumnName(i));
         }
         while (resultSet.next()) {
            Object[] rowData = new
Object[columnCount];
            for (int i = 0; i < columnCount;
i++) {
               rowData[i] =
resultSet.getObject(i + 1);
            }
            model.addRow(rowData);
         }
         resultSet.close();
         statement.close();
         connection.close();
      } catch (SQLException e) {
         e.printStackTrace();
      }
   }
   public static void main(String[] args) {
      SwingUtilities.invokeLater(new
Runnable() {
         @Override
         public void run() {
            new
ProjectTableDisplay().setVisible(true);
         }
      });
   }
}
```

Slip 13

Q1]

```
import java.sql.Connection;
import java.sql.DatabaseMetaData;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;

public class DatabaseInfo {
   public static final String DB_URL = "jdbc:mysql://localhost:3306/your_database";
   public static final String USER = "your_username";
   public static final String PASSWORD = "your_password";

   public static void main(String[] args) {
      try (Connection connection = DriverManager.getConnection(DB_URL, USER,
PASSWORD)) {
         DatabaseMetaData metaData = connection.getMetaData();

         System.out.println("Database Product Name: " +
metaData.getDatabaseProductName());
         System.out.println("Database Product Version: " +
metaData.getDatabaseProductVersion());
         System.out.println("Driver Name: " + metaData.getDriverName());
         System.out.println("Driver Version: " + metaData.getDriverVersion());
         System.out.println();

         ResultSet tablesResultSet = metaData.getTables(null, null, "%", new
String[]{"TABLE"});
         System.out.println("Tables in the database:");
         while (tablesResultSet.next()) {
            String tableName = tablesResultSet.getString("TABLE_NAME");
            System.out.println(tableName);
         }
         tablesResultSet.close();
      } catch (SQLException e) {
         e.printStackTrace();
      }
   }
}
```

Slip 14

Q1]

```
import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
public class SearchEngine {
   private static final String SEARCH_STRING = "your_search_string";
  public static void main(String[] args) {
      File currentDirectory = new File(".");
      File[] files = currentDirectory.listFiles();
      if (files != null) {
         for (File file : files) {
            if (file.isFile() && file.getName().endsWith(".txt")) {
               Thread searchThread = new Thread(new SearchRunnable(file));
               searchThread.start();
            }
         }
      }
   }
   static class SearchRunnable implements Runnable {
      private final File file;
      public SearchRunnable(File file) {
         this.file = file;
      }
      @Override
      public void run() {
         searchInFile();
      }
      private void searchInFile() {
         try (BufferedReader reader = new BufferedReader(new FileReader(file))) {
            String line;
            int lineNumber = 0;
            while ((line = reader.readLine()) != null) {
               lineNumber++;
               if (line.contains(SEARCH_STRING)) {
                  System.out.println("Found in file: " + file.getName() + " at line: " +
lineNumber);
               }
            }
         } catch (IOException e) {
            e.printStackTrace();
         }
      }    }    }
```

Slip 15

Q1]

```
public class ThreadInfoExample {
   public static void main(String[] args) {
      Thread thread = new Thread(new
MyRunnable());
      thread.setName("MyThread");

thread.setPriority(Thread.MAX_PRIORITY
);
      thread.start();

      System.out.println("Thread Name: "
+ thread.getName());
      System.out.println("Thread Priority:
" + thread.getPriority());
   }
}
class MyRunnable implements Runnable
{
   @Override
   public void run() {
      System.out.println("Thread is
running...");
   }
}
```

Q2]

```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import
javax.servlet.annotation.WebServlet;
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServlet;
import
javax.servlet.http.HttpServletRequest;
import
javax.servlet.http.HttpServletResponse;
@WebServlet("/VisitCounterServlet")
public class VisitCounterServlet extends
HttpServlet {
   private static final long
serialVersionUID = 1L;
   protected void
doGet(HttpServletRequest request,
HttpServletResponse response) throws
ServletException, IOException {

response.setContentType("text/html");
      PrintWriter out =
response.getWriter();
      Cookie[] cookies =
request.getCookies();
      int visitCount = 0;
      if (cookies != null) {
         for (Cookie cookie : cookies) {
            if
("visitCount".equals(cookie.getName())) {
               // If the cookie is found,
retrieve the value (visit count)
               visitCount =
Integer.parseInt(cookie.getValue());
               break;
            }
         }
      }
      visitCount++;

      Cookie visitCookie = new
Cookie("visitCount",
String.valueOf(visitCount));
      visitCookie.setMaxAge(24 * 60 * 60);
      response.addCookie(visitCookie);
      if (visitCount == 1) {
         out.println("<h2>Welcome to our
website!</h2>");
      } else {
         out.println("<h2>You have visited
this page " + visitCount + " times.</h2>");
      }
   }
}
```

Slip 16

Q1]

```
import java.util.TreeSet;

public class TreeSetExample {
    public static void main(String[] args) {
        TreeSet<String> colorsSet = new TreeSet<>();

        colorsSet.add("Red");
        colorsSet.add("Blue");
        colorsSet.add("Green");
        colorsSet.add("Yellow");
        colorsSet.add("Orange");
        System.out.println("Colors in ascending order:");
        for (String color : colorsSet) {
            System.out.println(color);
        }
    }
}
```

Slip 17

Q1]

```
import java.util.Scanner;
import java.util.Set;
import java.util.TreeSet;
public class SortedIntegers {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        Set<Integer> integersSet = new TreeSet<>();
        System.out.print("Enter the number of integers (N): ");
        int n = scanner.nextInt();
        System.out.println("Enter " + n + " integers:");
        for (int i = 0; i < n; i++) {
            int num = scanner.nextInt();
            integersSet.add(num);
        }
        System.out.println("Integers in sorted order (without duplicates):");
        for (int num : integersSet) {
            System.out.println(num);
        }
        scanner.close();
    }
}
```

Slip 18

Q1]

```
public class VowelPrinter {
    public static void main(String[] args) {
        String inputString = "Hello World";
        Thread vowelThread = new Thread(new VowelRunnable(inputString));
        vowelThread.start();
    }
}
class VowelRunnable implements Runnable {
    private final String inputString
    public VowelRunnable(String inputString) {
        this.inputString = inputString;
    }
    @Override
    public void run() {
        for (int i = 0; i < inputString.length(); i++) {
            char ch = inputString.charAt(i);
            if (isVowel(ch)) {
                System.out.println(ch); // Display the vowel
                try {
                    Thread.sleep(3000); // Sleep for 3 seconds
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
        }
    }
    private boolean isVowel(char ch) {
        ch = Character.toLowerCase(ch);
        return ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u';
    }
}
```

Slip 19

Q1]

```
import java.util.LinkedList;
import java.util.Scanner;

public class NegativeIntegers {
   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);

      // Create a LinkedList to store integers
      LinkedList<Integer> integersList = new LinkedList<>();

      // Accept 'N' integers from the user
      System.out.print("Enter the number of integers (N): ");
      int n = scanner.nextInt();

      System.out.println("Enter " + n + " integers:");
      for (int i = 0; i < n; i++) {
         int num = scanner.nextInt();
         integersList.add(num); // Add the integer to the LinkedList
      }

      // Display only the negative integers
      System.out.println("Negative integers:");
      for (int num : integersList) {
         if (num < 0) {
            System.out.println(num);
         }
      }

      scanner.close();
   }
}
```

Slip 20

Q2] import javax.swing.*;
import java.awt.*;

public class TempleDrawing extends JPanel implements Runnable {
  private int width;
  private int height;

  public TempleDrawing(int width, int height) {
    this.width = width;
    this.height = height;
  }

  @Override
  protected void paintComponent(Graphics g) {
    super.paintComponent(g);
    drawTemple(g);
  }

  private void drawTemple(Graphics g) {
    Graphics2D g2d = (Graphics2D) g;
    g2d.setColor(Color.WHITE);
    g2d.fillRect(0, 0, width, height);

    // Draw temple
    g2d.setColor(Color.GRAY);

    g2d.fillRect(width / 3, height / 4, width / 3, height / 2);

    g2d.setColor(Color.BLACK);
    g2d.fillRect(width / 3 + width / 12, height / 4 - height / 20, width / 6, height / 20);
    g2d.fillRect(width / 3 + width / 12, height * 3 / 4, width / 6, height / 20);
    g2d.fillRect(width / 3 - width / 40, height / 4, width / 20, height / 2);
    g2d.fillRect(width * 2 / 3 - width / 40, height / 4, width / 20, height / 2);
  }

  @Override
  public void run() {
    JFrame frame = new JFrame("Temple Drawing");

    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setSize(width, height);
    frame.setResizable(false);
    frame.add(this);
    frame.setVisible(true);
  }

  public static void main(String[] args) {
    int width = 400;
    int height = 400;

    TempleDrawing templeDrawing = new TempleDrawing(width, height);

    SwingUtilities.invokeLater(templeDrawing);
  }
}

Slip 21

Q1] import java.util.*;

```
public class SubjectNames {
   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);

      // Create a LinkedList to store subject names
      LinkedList<String> subjectList = new LinkedList<>();

      // Accept 'N' subject names from the user
      System.out.print("Enter the number of subjects (N): ");
      int n = scanner.nextInt();

      System.out.println("Enter " + n + " subject names:");
      for (int i = 0; i < n; i++) {
         String subjectName = scanner.next();
         subjectList.add(subjectName); // Add the subject name to the LinkedList
      }

      // Display subject names using Iterator
      System.out.println("Subject names:");
      Iterator<String> iterator = subjectList.iterator();
      while (iterator.hasNext()) {
         System.out.println(iterator.next());
      }

      scanner.close();
   }
}
```

Slip 22

Q1]

```
import java.util.Scanner;

public class
EmployeeManagementSystem {
  public static void main(String[] args) {
    Scanner scanner = new
Scanner(System.in);
    EmployeeDAO employeeDAO = new
EmployeeDAO();

    int choice;
    do {
      System.out.println("Menu:");
      System.out.println("1. Insert");
      System.out.println("2. Update");
      System.out.println("3. Display");
      System.out.println("4. Exit");
      System.out.print("Enter your
choice: ");
      choice = scanner.nextInt();

      switch (choice) {
        case 1:

employeeDAO.insertEmployee();
          break;
        case 2:

employeeDAO.updateEmployee();
          break;
        case 3:

employeeDAO.displayEmployees();
          break;
        case 4:
          System.out.println("Exiting
program...");
          break;
        default:
          System.out.println("Invalid
choice! Please enter a number between 1
and 4.");
      }
    } while (choice != 4);

    scanner.close();
  }
}

class EmployeeDAO {
  // Simulated database operations

  public void insertEmployee() {
    // Logic for inserting an employee
record
    System.out.println("Inserting an
employee...");
  }

  public void updateEmployee() {
    // Logic for updating an employee
record
    System.out.println("Updating an
employee...");
  }

  public void displayEmployees() {
    // Logic for displaying employee
records
    System.out.println("Displaying
employees...");
  }
}
```

Slip 23

Q1] import java.util.Scanner;

```
public class VowelDisplay {
    public static void main(String[] args) {
        Scanner scanner = new
Scanner(System.in);

        System.out.print("Enter a string: ");
        String inputString =
scanner.nextLine();

        VowelThread vowelThread = new
VowelThread(inputString);
        Thread thread = new
Thread(vowelThread);
        thread.start();

        scanner.close();
    }
}

class VowelThread implements Runnable
{
    private String inputString;

    public VowelThread(String inputString)
{
        this.inputString = inputString;
    }

    @Override
    public void run() {
        for (int i = 0; i < inputString.length();
i++) {
            char ch = inputString.charAt(i);
            if (isVowel(ch)) {
                System.out.println(ch); //
Display the vowel
                try {
                    Thread.sleep(3000); // Sleep
for 3 seconds
                } catch (InterruptedException e)
{
                    e.printStackTrace();
                }
            }
        }
    }

    // Method to check if a character is a
vowel
    private boolean isVowel(char ch) {
        ch = Character.toLowerCase(ch);
        return ch == 'a' || ch == 'e' || ch == 'i'
|| ch == 'o' || ch == 'u';
    }
}
```

Slip 24

Q1]
```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class TextScrolling extends JFrame implements Runnable {
    private JLabel label;
    private String text;
    private volatile boolean stop;

    public TextScrolling(String text) {
        this.text = text;
        label = new JLabel(text);
        label.setFont(new Font("Arial", Font.PLAIN, 20));
        add(label, BorderLayout.CENTER);
        setSize(400, 100);

        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setVisible(true);
    }

    @Override
    public void run() {
        int x = getWidth();
        while (!stop) {
            try {
                Thread.sleep(50);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
            x -= 5;
            if (x < -label.getWidth()) {
                x = getWidth();
            }
            label.setLocation(x, 0);
        }
    }

    public void stopScrolling() {
        stop = true;
    }

    public static void main(String[] args) {
        TextScrolling scrollingText = new TextScrolling("Hello, World!");
        Thread thread = new Thread(scrollingText);
        thread.start();


        scrollingText.addWindowListener(new WindowAdapter() {
            @Override
            public void windowClosing(WindowEvent e) {
                scrollingText.stopScrolling();
            }
        });
    }
}
```

Slip 25

Q1] html :- <!DOCTYPE html>

```
<html>
<head>
  <meta charset="UTF-8">
  <title>Voter Eligibility Checker</title>
</head>
<body>
  <h2>Voter Eligibility Checker</h2>
  <form action="CheckEligibility.jsp" method="post">
    <label for="name">Name:</label>
    <input type="text" id="name" name="name" required><br><br>
    <label for="age">Age:</label>
    <input type="number" id="age" name="age" required><br><br>
    <input type="submit" value="Check Eligibility">
  </form>
</body>
</html>
```

Jsp :-

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Voter Eligibility Result</title>
</head>
<body>
  <h2>Voter Eligibility Result</h2>
  <%
    String name = request.getParameter("name");
    int age = Integer.parseInt(request.getParameter("age"));

    if (age >= 18) {
  %>
    <p><%= name %> is eligible to vote.</p>
  <% } else { %>
    <p><%= name %> is not eligible to vote.</p>
  <% } %>
</body>
</html>
```

Slip 26

Q1] import java.sql.*;

```
public class DeleteEmployee {
   // JDBC URL, username, and password
   public static final String DB_URL =
"jdbc:mysql://localhost:3306/your_datab
ase";
   public static final String USER =
"your_username";
   public static final String PASSWORD =
"your_password";

   public static void main(String[] args) {
      if (args.length != 1) {
         System.out.println("Usage: java
DeleteEmployee <employee_id>");
         return;
      }

      int employeeId =
Integer.parseInt(args[0]);

      Connection connection = null;
      PreparedStatement
preparedStatement = null;

      try {
         // Establish connection to the
database
         connection =
DriverManager.getConnection(DB_URL,
USER, PASSWORD);

         // Prepare SQL statement to delete
employee details
         String sql = "DELETE FROM
employee WHERE ENo = ?";
         preparedStatement =
connection.prepareStatement(sql);

         // Set parameters for
PreparedStatement
         preparedStatement.setInt(1,
employeeId);

         // Execute the delete operation
         int rowsDeleted =
preparedStatement.executeUpdate();
         if (rowsDeleted > 0) {
            System.out.println("Employee
details deleted successfully.");
         } else {
            System.out.println("No
employee found with ID: " +
employeeId);
         }

      } catch (SQLException e) {
         e.printStackTrace();
      } finally {
         // Close resources in finally block
         try {
            if (preparedStatement != null)
preparedStatement.close();
            if (connection != null)
connection.close();
         } catch (SQLException e) {
            e.printStackTrace();
         }
      }
   }
}
```

Slip 27

Q1] import javax.swing.*;

```
import java.awt.*;
import java.sql.*;

public class CollegeDetails extends JFrame {
   private JTable table;
   public CollegeDetails() {
      setTitle("College Details");
      setSize(600, 400);
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      JPanel panel = new JPanel();
      getContentPane().add(panel, BorderLayout.CENTER);
      JScrollPane scrollPane = new JScrollPane();
      panel.add(scrollPane);
      table = new JTable();
      scrollPane.setViewportView(table);

      loadData();
   }
   private void loadData() {
      Connection connection = null;
      Statement statement = null;
      ResultSet resultSet = null;

      try {
         // Establish connection to the database
         connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/your_database", "your_username", "your_password");

         // Create SQL statement to select all records from College table
         statement = connection.createStatement();
         String query = "SELECT * FROM College";
         resultSet = statement.executeQuery(query);

         // Create a TableModel to hold the data
         DefaultTableModel model = new DefaultTableModel();
         table.setModel(model);
         model.addColumn("CID");
         model.addColumn("CName");
         model.addColumn("Address");
         model.addColumn("Year");
         while (resultSet.next()) {
            Object[] row = {resultSet.getInt("CID"), resultSet.getString("CName"),

resultSet.getString("address"), resultSet.getInt("Year")};
            model.addRow(row);
         }
      } catch (SQLException e) {
         e.printStackTrace();
      } finally {
         try {
            if (resultSet != null) resultSet.close();
            if (statement != null) statement.close();
            if (connection != null) connection.close();
         } catch (SQLException e) {
            e.printStackTrace();
         }
      }
   }
   public static void main(String[] args) {
      SwingUtilities.invokeLater(() -> {
         CollegeDetails collegeDetails = new CollegeDetails();
         collegeDetails.setVisible(true);
      });
   }
}
```

Slip 28

Q2]
```
public class CurrentThreadName {
    public static void main(String[] args) {
        // Create and start a new thread
        Thread thread = new Thread(new MyRunnable());
        thread.start();

        // Display the name of the main thread
        System.out.println("Main thread name: " + Thread.currentThread().getName());
    }
}

class MyRunnable implements Runnable {
    @Override
    public void run() {
        // Display the name of the currently executing thread
        System.out.println("Currently executing thread name: " +
Thread.currentThread().getName());
    }
}
```

Slip 29

Q2]
```
import java.util.LinkedList;
public class LinkedListOperations {
    public static void main(String[] args) {
        LinkedList<Integer> linkedList = new LinkedList<>();
        linkedList.addFirst(10);
        linkedList.addFirst(20);
        linkedList.addFirst(30);
        System.out.println("LinkedList after adding elements at first position: " + linkedList);
        if (!linkedList.isEmpty()) {
            linkedList.removeLast();
            System.out.println("LinkedList after deleting last element: " + linkedList);
        } else {
            System.out.println("LinkedList is empty. Cannot delete last element.");
        }
        System.out.println("Size of LinkedList: " + linkedList.size());
    }
}
```

Slip 30

Q1]

```
import java.sql.*;

public class ScrollableResultSetExample {
   public static void main(String[] args) {
      Connection connection = null;
      Statement statement = null;
      ResultSet resultSet = null;
      try {
         connection =
DriverManager.getConnection("jdbc:mys
ql://localhost:3306/your_database",
"your_username", "your_password");

         statement =
connection.createStatement(ResultSet.TY
PE_SCROLL_SENSITIVE,
ResultSet.CONCUR_READ_ONLY);
         resultSet =
statement.executeQuery("SELECT *
FROM Teacher");
         resultSet.beforeFirst();


System.out.println("TID\tTName\tSalary
");
         while (resultSet.next()) {
            int tid = resultSet.getInt("TID");
            String tname =
resultSet.getString("TName");
            double salary =
resultSet.getDouble("Salary");
            System.out.println(tid + "\t" +
tname + "\t" + salary);
         }
         resultSet.afterLast();
         System.out.println("\nDisplaying
data in reverse order:");
         while (resultSet.previous()) {
            int tid = resultSet.getInt("TID");
            String tname =
resultSet.getString("TName");
            double salary =
resultSet.getDouble("Salary");
            System.out.println(tid + "\t" +
tname + "\t" + salary);
         }
      } catch (SQLException e) {
         e.printStackTrace();
      } finally {
         try {
            if (resultSet != null)
resultSet.close();
            if (statement != null)
statement.close();
            if (connection != null)
connection.close();
         } catch (SQLException e) {
            e.printStackTrace();
         }
      }
   }
}
```